

CCCCCCCCCCCC C LLL
CCCCCCCCCCCC C LLL
CCCCCCCCCCCC C LLL

CCC
CCCCCCCCCCCC LLLL
CCCCCCCCCCCC LLLL
CCCCCCCCCCCC LLLL

••FILE••ID••RUNDET

K 12

```

RRRRRRRRR   UU      UU  NN      NN  DDDDDDDDD  EEEEEEEEEE  TTTTTTTTTT
RRRRRRRRR   UU      UU  NN      NN  DDDDDDDDD  EEEEEEEEEE  TTTTTTTTTT
RR      RR  UU      UU  NN      NN  DD      DD  EE      EE  TT
RR      RR  UU      UU  NN      NN  DD      DD  EE      EE  TT
RR      RR  UU      UU  NNNN    NN  DD      DD  EE      EE  TT
RR      RR  UU      UU  NNNN    NN  DD      DD  EE      EE  TT
RRRRRRRRR   UU      UU  NN  NN  NN  DD      DD  EEEEEEEEEE  TT
RRRRRRRRR   UU      UU  NN  NN  NN  DD      DD  EEEEEEEEEE  TT
RR      RR  UU      UU  NN      NNNN  DD      DD  EE      EE  TT
RR      RR  UU      UU  NN      NNNN  DD      DD  EE      EE  TT
RR      RR  UU      UU  NN      NN  DD      DD  EE      EE  TT
RR      RR  UU      UU  NN      NN  DD      DD  EE      EE  TT
RR      RR  UU      UU  UUUUUUUUUU  NN      NN  DDDDDDDDD  EEEEEEEEEE  TT
RR      RR  UU      UU  UUUUUUUUUU  NN      NN  DDDDDDDDD  EEEEEEEEEE  TT

```

The diagram illustrates two sets of binary strings, each consisting of 11 strings of length 11.

Left Set: All 11 strings consist entirely of the character 'L'.

Right Set: All 11 strings consist entirely of the character 'S'.

```
1 0001 0 MODULE rundet (%TITLE 'Run Detached Process -- [LI Utility Procedure'  
2 0002 0 IDENT = 'V04-000',  
3 0003 0 MAIN = run_detached) =  
4 0004 1 BEGIN  
5 0005 1 *****  
6 0006 1 *  
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
9 0009 1 * ALL RIGHTS RESERVED.  
10 0010 1 *  
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
16 0016 1 * TRANSFERRED.  
17 0017 1 *  
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
20 0020 1 * CORPORATION.  
21 0021 1 *  
22 0022 1 *  
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
25 0025 1 *  
26 0026 1 *  
27 0027 1 *****  
28 0028 1 ++  
29 0029 1  
30 0030 1 Facility:  
31 0031 1  
32 0032 1 CLI Utility  
33 0033 1  
34 0034 1 Abstract:  
35 0035 1  
36 0036 1  
37 0037 1 This module contains the routines necessary to act as a CLI interface  
38 0038 1 to the create process ($CREPRC) and schedule wakeup ($SCHEDWK) system  
39 0039 1 services.  
40 0040 1 Environment:  
41 0041 1  
42 0042 1 VAX/VMS User Mode, Non-Privileged  
43 0043 1  
44 0044 1  
45 0045 1 Author:  
46 0046 1 Michael T. Rhodes. Creation Date: March, 1983  
47 0047 1  
48 0048 1  
49 0049 1 Modified By:  
50 0050 1  
51 0051 1 V03-007 MCN0159 Maria del C. Nasr 28-Mar-1984  
52 0052 1 Use LIB$TPARSE to parse uic instead of RUN_CVTUIC.  
53 0053 1  
54 0054 1 V03-006 RAS0274 Ron Schaefer 20-Mar-1984  
55 0055 1 Make this work with searchlists by using LIB$FIND FILE  
56 0056 1 to parse the image filespec. Wildcards are not allowed.  
57 0057 1
```

58	0058	1	V03-005	TMK0001	Todd M. Katz	13-Nov-1983
59	0059	1			Add the qualifier /JOB_TABLE_QUOTA. The use of this	
60	0060	1			qualifier allows the creator of a detached process to	
61	0061	1			specify its job-wide logical name table creation quota.	
62	0062	1				
63	0063	1	V03-004	MTR0002	Michael T. Rhodes	22-Jul-1983
64	0064	1			Correct qualifier name /PROCESS to /PROCESS_NAME.	
65	0065	1				
66	0066	1	V03-003	MTR0001	Michael T. Rhodes	29-Apr-1983
67	0067	1			Convert privilege processing to use common CLI utility	
68	0068	1			routine PRVSSETPRIV. Also change PID message to PROC_ID.	
69	0069	1				
70	0070	1	V03-002	WMC0002	Wayne Cardoza	14-Apr-1983
71	0071	1			Qualifier is /DETACHED	
72	0072	1				
73	0073	1	V03-001	WMC0001	Wayne Cardoza	11-Apr-1983
74	0074	1			Add /DETACH and /DUMP flags.	
75	0075	1				
76	0076	1	--			
77	0077	1				

```

79      0078 1 %SBTTL 'Declarations'
80      0079 1
81      0080 1 | Include Files:
82      0081 1
83      0082 1 | LIBRARY 'SYSSLIBRARY:LIB';
84      0083 1 | LIBRARY 'SYSSLIBRARY:TPAMAC';
85      0084 1
86      0085 1
87      0086 1 | Table of Contents:
88      0087 1
89      0088 1 FORWARD ROUTINE
90      0089 1   run_detached : NOVALUE,
91      0090 1   init_arg_list : NOVALUE,
92      0091 1   parse_image_spec: NOVALUE,
93      0092 1   get_wakeup_info : NOVALUE,
94      0093 1   get_uic : NOVALUE,
95      0094 1   get_privileges : NOVALUE,
96      0095 1   get_quotas : NOVALUE,
97      0096 1   get_cplm : NOVALUE,
98      0097 1   get_value : NOVALUE,
99      0098 1   insert_quota : NOVALUE,
100     0099 1   get_stsflgs : NOVALUE,
101     0100 1   schedule_process: NOVALUE;
102     0101 1
103     0102 1
104     0103 1 | External references:
105     0104 1
106     0105 1 EXTERNAL ROUTINE
107     0106 1   CLISGET_VALUE : ADDRESSING_MODE (GENERAL),
108     0107 1   CLISPRESNT : ADDRESSING_MODE (GENERAL),
109     0108 1   lib$cvt_dtime : ADDRESSING_MODE (GENERAL),
110     0109 1   LIB$CVT_DX_DX : ADDRESSING_MODE (GENERAL),
111     0110 1   lib$cvt_time : ADDRESSING_MODE (GENERAL),
112     0111 1   LIB$GET_VM : ADDRESSING_MODE (GENERAL),
113     0112 1   LIB$FIND_FILE : ADDRESSING_MODE (GENERAL),
114     0113 1   LIB$TPARSE : ADDRESSING_MODE (GENERAL),
115     0114 1   prv$setpriv : ADDRESSING_MODE (GENERAL);
116     0115 1
117     0116 1
118     0117 1 | Define message codes...
119     0118 1
120     P 0119 1 $SHR_MSGDEF (RUN,192,GLOBAL,
121     P 0120 1   (INSVIRMEM,SEVERE),
122     P 0121 1   (INVQUAVAL,ERROR),
123     P 0122 1   (PARSEFAIL,SEVERE),
124     P 0123 1   (SYNTAX,SEVERE));
125     0124 1
126     0125 1 EXTERNAL LITERAL
127     0126 1   cli$_negated,
128     0127 1   run$_creprc,
129     0128 1   run$_proc_id,
130     0129 1   run$_getjpi,
131     0130 1   run$_cvterr,
132     0131 1   run$_invuic,
133     0132 1   run$_lval,
134     0133 1   run$_schdwk;
135     0134 1

```

| Define VMS structures.
| T\$PARSE structures

| Create a sub or detached process.
| Initialize the SCREPRC and SSCHDWK argument lists.
| Obtain the expanded image file specification.
| Get info for the SSCHDWK .
| Get the UIC value.
| Set up the privilege mask vector for the process.
| Set up the quota list for the process.
| Special case for CPU time limit quota.
| Get the value of the current command line entity.
| Insert a quota list entry.
| Set up the initial process status flag vector.
| Schedule the process to be executed.

| CLI call back routine to get command line entity.
| CLI call back routine to determine entity presence
| Private CLI routine to convert a string to a delta
| General conversion routine.
| Private CLI routine to convert a string to an abso
| Library routine to obtain virtual memory.
| Library routine to parse filespecs.
| Table driven parser
| Private CLI routine to process privileges and set

| Define the shareable messages first.
| Insufficient virtual memory.
| Invalid qualifier value.
| Error parsing file spec.
| Error parsing command entity.

| Command line entity was explicitly negated.
| Create process failed.
| Identification of created process.
| Error obtaining job and process information.
| Error converting entity value.
| Invalid UIC.
| Illegal CPU time limit.
| Failed to schedule the wakeup request.

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure
Declarations

B 13
16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.B32;1

; 136

0135 1

Page 4
(2)

```

138 0136 1 %SBTTL 'Declarations -- Private Storage'
139 0137 1
140 0138 1 Command qualifiers, keywords, and keyword paths
141 0139 1
142 0140 1 BIND
143 0141 1     accounting = $descriptor ('ACCOUNTING'),
144 0142 1     authorize = $descriptor ('AUTHORIZE'),
145 0143 1     delay = $descriptor ('DELAY'),
146 0144 1     detach = $descriptor ('DETACHED'),
147 0145 1     dump = $descriptor ('DUMP'),
148 0146 1     error = $descriptor ('ERROR'),
149 0147 1     input = $descriptor ('INPUT'),
150 0148 1     interval = $descriptor ('INTERVAL'),
151 0149 1     mailbox = $descriptor ('MAILBOX'),
152 0150 1     output = $descriptor ('OUTPUT'),
153 0151 1     p1 = $descriptor ('P1'),
154 0152 1     priority = $descriptor ('PRIORITY'),
155 0153 1     privileges = $descriptor ('PRIVILEGES'),
156 0154 1     process = $descriptor ('PROCESS NAME'),
157 0155 1     resource_wait = $descriptor ('RESOURCE WAIT'),
158 0156 1     schedule = $descriptor ('SCHEDULE'),
159 0157 1     service_fail = $descriptor ('SERVICE FAILURE'),
160 0158 1     swapping = $descriptor ('SWAPPING'),
161 0159 1     uic = $descriptor ('UIC');
162 0160 1
163 0161 1 LITERAL
164 0162 1     true = 1, false = 0,
165 0163 1     jpientries = 3,
166 0164 1     jpilistsize = jpientries * 12,
167 0165 1     list_k_entry_size = 5,
168 0166 1     priv_entries = 31;
169 0167 1
170 0168 1 MACRO
171 0169 1     list_entry (name, entity) = BYTE (name), LONG ($descriptor (entity)) %,
172 0170 1     list_b_name = 0,0,8,0 %,
173 0171 1     list_l_value = 1,0,32,0 %;
174 0172 1
175 0173 1 OWN
176 0174 1     run$sa_error : $bblock [dsc$e_s_bln],
177 0175 1     run$sa_image : $bblock [dsc$e_s_bln],
178 0176 1     run$sa_input : $bblock [dsc$e_s_bln],
179 0177 1     run$sa_input_desc: $bblock [dsc$e_s_bln],
180 0178 1     run$sa_output : $bblock [dsc$e_s_bln],
181 0179 1     run$sa_prcnam : $bblock [dsc$e_s_bln],
182 0180 1     run$sa_quota,
183 0181 1     run$l_baspri,
184 0182 1     run$l_mbxit,
185 0183 1     run$l_pid,
186 0184 1     run$l_status,
187 0185 1     run$l_stsflg : $bblock [4],
188 0186 1     run$l_uic,
189 0187 1     run$g_daytim : VECTOR [2, LONG] INITIAL (-1,-1),
190 0188 1     run$g_interval : VECTOR [2, LONG] INITIAL (0,0),
191 0189 1     run$g_prvadr : VECTOR [2, LONG],
192 0190 1     run$sa_image_buf : $bblock [namSc_maxrss],
193 0191 1     run$sa_findfile : ref $bblock,
194 0192 1

```

Log accounting records for created process.
 Perform user authorization when image is loaded.
 Hibernate process and awaken after delta t.
 Detached process.
 Image dump requested.
 Error device (SYS\$ERROR).
 Input device (SYS\$INPUT).
 Hibernate process and awaken at regularly.
 Unit number of termination mailbox for this.
 Output device (SYS\$OUTPUT).
 Image file specification parameter.
 Base priority at which the created process.
 Defines the privileges for the created process.
 Specifies the process name.
 Enable/disables resource wait mode for the process.
 Hibernate the process and awaken at absolute time.
 Enable/disable system service failure exception.
 Enable/disable process swapping.
 Detached process UIC.

Boolean opcodes.
 Number of entries in the \$GETJPI item list.
 Number of bytes required for the \$GETJPI item list.
 Number of bytes in a list entry.
 Number of real privileges (prv\$v_xxx).

Macro to create list entries for the quota.
 Name field access formal.
 Value field access formal.

Descriptor for error device specification.
 Descriptor for image file specification.
 Descriptor for input device specification.
 General purpose dynamic input descriptor.
 Descriptor for output device specification.
 Descriptor for process name.
 Quota list head address.
 Base execution priority for the process.
 Termination mailbox unit number.
 Created process' PID.
 Global Status vector.
 Initial process state status flags.
 Detached process' UIC.
 Time at which the process is to be awakened.
 Interval at which the wake up request is issued.
 Privilege vector.
 Buffer to hold the expanded file specification.
 Context ptr for LIB\$FIND_FILE.

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure Data Structures -- \$GETJPI Item list

D 13
16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CL|UTL.SRC]RUNDET.932:1

Page 6
(4)

```

196 0193 1 %SBTTL 'Data Structures -- $GETJPI Item List'
197 0194 1 ++
198 0195 1
199 0196 1 Functional Description:
200 0197 1
201 0198 1 This structure is used to obtain default job and process information
202 0199 1 for the created process.
203 0200 1
204 0201 1 --
205 0202 1 run$sa_getjpi : $bbblock [jpilistsize] ! Item list for $GETJPI.
206 0203 1 INITIAL (
207 0204 1 WORD (4, jpis_prib), LONG (run$1_baspri, 0),
208 0205 1 WORD (8, jpis_procpri), LONG (run$q_prvadr, 0),
209 0206 1 WORD (0, 0), LONG (0, 0)),
210 0207 1 ! Process base priority.
211 ! Process 'SAME' privileges.
212 ! Item list terminator.

```

```
212 0208 1 %SBTTL 'Data Structures -- Quota Table'  
213 0209 1 ++  
214 0210 1  
215 0211 1 Functional Description:  
216 0212 1  
217 0213 1 This structure is used to parse and process the command line entries  
218 0214 1 which comprise the quota list values and to establish the initial quota  
219 0215 1 context for the SCREPRC system service.  
220 0216 1  
221 0217 1 --  
222 0218 1 run$a_quota_tbl : Sbblock [(pql$_length - 1) * list_k_entry_size] ! Quota table.  
223 0219 1 INITIAL (  
224 0220 1     list_entry (pql$_astlm,      'AST LIMIT'),  
225 0221 1     list_entry (pql$_bytlm,    'BUFFER LIMIT'),  
226 0222 1     list_entry (pql$_enqlm,    'ENQUEUE LIMIT'),  
227 0223 1     list_entry (pql$_wsextent, 'EXTENT'),  
228 0224 1     list_entry (pql$_fillm,     'FILE LIMIT'),  
229 0225 1     list_entry (pql$_biolm,    'IO_BUFFERED'),  
230 0226 1     list_entry (pql$_diolm,    'IO_DIRECT'),  
231 0227 1     list_entry (pql$_wsquota,   'MAXIMUM WORKING_SET'),  
232 0228 1     list_entry (pql$_pgflquota, 'PAGE FILE'),  
233 0229 1     list_entry (pql$_tqelm,     'QUEUE LIMIT'),  
234 0230 1     list_entry (pql$_prclm,     'SUBPROCESS LIMIT'),  
235 0231 1     list_entry (pql$_cpulm,     'TIME LIMIT'),  
236 0232 1     list_entry (pql$_wsdefault, 'WORKING SET'),  
237 0233 1     list_entry (pql$_jtquota,   'JOB_TABLE_QUOTA')  
238 0234 1 );  
239 0235 1  
240 0236 1 ! TPARSE state table to parse the uic value  
241 0237 1  
242 0238 1  
243 0239 1 $INIT_STATE ( uic_states, uic_keys );  
244 0240 1  
P 0241 1 $STATE ( .  
P 0242 1     (tpa$_ident,...,run$1_uic)  
P 0243 1     );  
P 0244 1  
P 0245 1 $STATE ( .  
P 0246 1     (tpa$_eos, tpa$_exit)  
P 0247 1     );
```

```
253      0248 1 %SBTTL 'run_detached -- Create a sub or detached process'  
254      0249 1 ROUTINE run_detached : NOVALUE =  
255      0250 1 ++  
256      0251 1 Functional Description:  
257      0252 1 This routine is responsible for calling the initialization procedure  
258      0253 1 which will build the argument lists used for performing the create  
259      0254 1 process ($CREPRC) system service. If the process is to be scheduled  
260      0255 1 to be awakened or re-executed at specific intervals, a call is made  
261      0256 1 to perform that action.  
262      0257 1  
263      0258 1  
264      0259 1 Implicit Inputs:  
265      0260 1 The command line supplied by the CLI.  
266      0261 1  
267      0262 1 Implicit Outputs:  
268      0263 1 The process is created with the process identification (PID) of the  
269      0264 1 created process SIGNALLED to the caller as a success status message.  
270      0265 1  
271      0266 1 Routine Value:  
272      0267 1  
273      0268 1 SSS_NORMAL Process was created/scheduled without errors.  
274      0269 1  
275      0270 1  
276      0271 1 false Error status from failed routine.  
277      0272 1  
278      0273 1  
279      0274 1 Side Effects:  
280      0275 1 The process may have been hibernated with a scheduled wake up pending.  
281      0276 1  
282      0277 1  
283      0278 1  
284      0279 1 --  
285      0280 2 BEGIN  
286      0281 2  
287      0282 2 init_arg_list (); ! Initialize the argument lists for the $CREPRC and  
288      0283 2  
P 0284 3 IF NOT (run$1_status = $CREPRC (PIDADR = run$1_pid,  
P 0285 3 IMAGE = run$1_image,  
P 0286 3 INPUT = run$1_input,  
P 0287 3 OUTPUT = run$1_output,  
P 0288 3 ERROR = run$1_error,  
P 0289 3 PRVADR = run$1_prvadr,  
P 0290 3 QUOTA = .run$1_quota,  
P 0291 3 PRCNAM = run$1_prcnam,  
P 0292 3 BASPRI = .run$1_baspri,  
P 0293 3 UIC = .run$1_uic,  
P 0294 3 MBXUNT = .run$1_mbxunt,  
P 0295 3 STSFLG = .run$1_stsflg)); ! Process ID of created process.  
P 0296 3 ! File specification of the image to execute.  
P 0297 3 ! Input device specification (assigned to SYSSINPUT)  
P 0298 3 ! Output device specification (assigned to SYSSOUTPUT)  
P 0299 3 ! Error device specification (assigned to SYS$ERROR)  
P 0300 3 ! Quadword privilege vector address.  
P 0301 3 ! List head address for the quota list.  
P 0302 3 ! Process name.  
P 0303 3 ! Process base execution priority.  
P 0304 3 ! Process UIC.  
P 0305 3 ! Termination mailbox unit number.  
P 0306 3 ! Initial process state status flags.  
P 0307 3  
P 0308 3 THEN ! Any errors, stop here and inform the user.  
P 0309 3 SIGNAL_STOP (run$1_creprc, 0, .run$1_status)  
P 0310 3 ELSE ! Process was created successfully, show it's PID to  
P 0311 3 SIGNAL (run$1_proc_id, 1, .run$1_pid); ! Should we leave a wake up call?  
P 0312 3  
P 0313 3 IF .run$1_stsflg [prcv_hiber]  
P 0314 3 THEN schedule_process();  
P 0315 3  
P 0316 3 END; ! of ROUTINE run_detached
```

; .TITLE RUNDET Run Detached Process -- CLI Utility Proc
; .IDENT \V04-000\ edure
; .PSECT _LIBSSTATES,NOWRT, SHR, PIC,1
; 00000 UIC_STATES::
; 4SEC 00000 ;TPASTYPE :BLKB 0
; 00000000* 00002 ;TPASADDR U.2: WORD 17900 ;
; 15F7 00006 ;TPASTYPE U.3: LONG <<RUNSL_UIC-U.3>-4> ;
; FFFF 00008 ;TPASTARGET U.4: WORD 5623 ;
; U.5: WORD -1 ;
; .PSECT _LIBSKEYOS,NOWRT, SHR, PIC,1
; 00000 UIC_KEYS::
; 00000 ;TPASKEYO :BLKB 0
; U.1: .BLKB 0
; .PSECT SPLITS,NOWRT,NOEXE,2
; 47 4E 49 54 4E 55 4F 43 43 41 00000 P.AAB: .ASCII \ACCOUNTING\
; 0000A :BLKB 2 ;
; 0000000A 0000C P.AAA: .LONG 10 ;
; 00000000* 00010 :ADDRESS P.AAB ;
; 45 5A 49 52 4F 48 54 55 41 00014 P.AAD: .ASCII \AUTHORIZE\
; 0001D :BLKB 3 ;
; 00000009 00020 P.AAC: .LONG 9 ;
; 00000000* 00024 :ADDRESS P.AAD ;
; 59 41 4C 45 44 00028 P.AAF: .ASCII \DELAY\
; 0002D :BLKB 3 ;
; 00000005 00030 P.AAE: .LONG 5 ;
; 00000000* 00034 :ADDRESS P.AAF ;
; 44 45 48 43 41 54 45 44 00038 P.AAH: .ASCII \DETACHED\
; 00000008 00040 P.AAG: .LONG 8 ;
; 00000000* 00044 :ADDRESS P.AAH ;
; 50 4D 55 44 00048 P.AAJ: .ASCII \DUMP\
; 00000004 0004C P.AAI: .LONG 4 ;
; 00000000* 00050 :ADDRESS P.AAJ ;
; 52 4F 52 52 45 00054 P.AAL: .ASCII \ERROR\
; 00059 :BLKB 3 ;
; 00000005 0005C P.AAK: .LONG 5 ;
; 00000000* 00060 :ADDRESS P.AAL ;
; 54 55 50 4E 49 00064 P.AAN: .ASCII \INPUT\
; 00069 :BLKB 3 ;
; 00000005 0006C P.AAM: .LONG 5 ;
; 00000000* 00070 :ADDRESS P.AAN ;
; 4C 41 56 52 45 54 4E 49 00074 P.AAP: .ASCII \INTERVAL\
; 00000008 0007C P.AAO: .LONG 8 ;
; 00000000* 00080 :ADDRESS P.AAP ;

58 4F 42 4C 49 41 4D 00084 P.AAR: .ASCII \MAILBOX\
0008B :BLKB 1
00000007 0008C P.AAQ: .LONG 7
00000000, 00090 :ADDRESS P.AAR
54 55 50 54 55 4F 00094 P.AAT: .ASCII \OUTPUT\
0009A :BLKB 2
00000006 0009C P.AAS: .LONG 6
00000000, 000A0 :ADDRESS P.AAT
31 50 000A4 P.AAV: .ASCII \PI\
000A6 :BLKB 2
00000002, 000A8 P.AAU: .LONG 2
00000000, 000AC :ADDRESS P.AAV
59 54 49 52 4F 49 52 50 000B0 P.AAX: .ASCII \PRIORITY\
00000008 000B8 P.AAW: .LONG 8
00000000, 000BC :ADDRESS P.AAX
53 45 47 45 4C 49 56 49 52 50 000C0 P.AAZ: .ASCII \PRIVILEGES\
000CA :BLKB 2
0000000A 000CC P.AAY: .LONG 10
00000000, 000D0 :ADDRESS P.AAZ
45 4D 41 4E 5F 53 53 45 43 4F 52 50 000D4 P.ABB: .ASCII \PROCESS_NAME\
0000000C 000E0 P.ABA: .LONG 12
00000000, 000E4 :ADDRESS P.ABB
54 49 41 57 5F 45 43 52 55 4F 53 45 52 000E8 P.ABD: .ASCII \RESOURCE_WAIT\
000F5 :BLKB 3
0000000D 000F8 P.ABC: .LONG 13
00000000, 000FC :ADDRESS P.ABD
45 4C 55 44 45 48 43 53 00100 P.ABF: .ASCII \SCHEDULE\
00000008 00108 P.ABE: .LONG 8
00000000, 0010C :ADDRESS P.ABF
45 52 55 4C 49 41 46 5F 45 43 49 56 52 45 53 00110 P.ABH: .ASCII \SERVICE_FAILURE\
0011F :BLKB 1
0000000F 00120 P.ABG: .LONG 15
00000000, 00124 :ADDRESS P.ABH
47 4E 49 50 50 41 57 53 00128 P.ABJ: .ASCII \SWAPPING\
00000008 00130 P.ABI: .LONG 8
00000000, 00134 :ADDRESS P.ABJ
43 49 55 00138 P.ABL: .ASCII \UIC\
0013B :BLKB 1
00000003 0013C P.ABK: .LONG 3
00000000, 00140 :ADDRESS P.ABL
54 49 4D 49 4C 5F 54 53 41 00144 P.ABN: .ASCII \AST_LIMIT\
0014D :BLKB 3
00000009 00150 P.ABM: .LONG 9
00000000, 00154 :ADDRESS P.ABN
54 49 4D 49 4C 5F 52 45 46 46 55 42 00158 P.ABP: .ASCII \BUFFER_LIMIT\
0000000C 00164 P.ABO: .LONG 12
00000000, 00168 :ADDRESS P.ABP
54 49 4D 49 4C 5F 45 55 51 4E 45 0016C P.ABR: .ASCII \ENQUEUE_LIMIT\
00179 :BLKB 3
0000000D, 0017C P.ABQ: .LONG 13
00000000, 00180 :ADDRESS P.ABR
54 4E 45 54 58 45 00184 P.ABT: .ASCII \EXTENT\
0018A :BLKB 2
00000006, 0018C P.ABS: .LONG 6
00000000, 00190 :ADDRESS P.ABT
54 49 4D 49 4C 5F 45 4C 49 46 00194 P.ABV: .ASCII \FILE_LIMIT\
0019E :BLKB 2

RUNDET
V04-000

I 13
Run Detached Process -- CLI Utility Procedure 16-Sep-1984 00:27:00 VAX-11 Bliss-32 v4.0-742
run_detached -- Create a sub or detached process 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32;1

Page 11
(6)

44 45 52 45 46 46 55 42 SF 4F 49 0000000A 001A0 P.ABU: .LONG 10
00000000 001A4 :ADDRESS P.ABV
001B3 :ASCII \IO_BUFFERED\
54 43 45 52 49 44 SF 4F 49 0000000B 001B4 P.ABW: .LONG 11
00000000 001B8 :ADDRESS P.ABX
001C5 :ASCII \IO_DIRECT\
001C8 P.ABY: .LONG 9
00000009 001CC :ADDRESS P.ABZ
00000000 001DF :ASCII \MAXIMUM_WORKING_SET\
47 4E 49 48 52 4F 57 SF 4D 55 4D 49 58 41 4D 001E3 P.ACA: .BLKB 1
54 45 53 53 45 43 4F 52 50 42 55 53 54 001E4 P.ACD: .LONG 19
00000013 001E8 :ADDRESS P.ACB
001F5 :ASCII \PAGE_FILE\
54 49 4D 49 4C 5F 45 55 45 55 51 001F8 P.ACC: .LONG 9
00000009 001FC :ADDRESS P.ACD
00200 :ASCII \QUEUE_LIMIT\
45 4C 49 46 SF 45 47 41 50 0020B P.ACF: .BLKB 1
0000000B 0020C P.ACE: .LONG 11
00000000 00210 :ADDRESS P.ACF
49 4D 49 4C SF 53 53 45 43 4F 52 50 42 55 53 00214 P.ACH: .ASCII \SUBPROCESS_LIMIT\
00223 :ADDRESS P.ACJ
00000010 00224 P.ACG: .LONG 16
00000000 00228 :ADDRESS P.ACH
54 49 4D 49 4C 5F 45 4D 49 54 00230 P.ACJ: .ASCII \TIME_LIMIT\
00236 :BLKB 2
0000000A 00238 P.ACI: .LONG 10
00000000 0023C :ADDRESS P.ACJ
54 45 53 5F 47 4E 49 4B 52 4F 57 00240 P.ACL: .ASCII \WORKING_SET\
00248 :BLKB 1
0000000B 0024C P.ACK: .LONG 11
00000000 00250 :ADDRESS P.ACL
41 54 4F 55 51 SF 45 4C 42 41 54 5F 42 4F 4A 00254 P.ACN: .ASCII \JOB_TABLE_QUOTA\
00263 :BLKB 1
0000000F 00264 P.ACML: .LONG 15
00000000 00268 :ADDRESS P.ACN
00000000 .PSECT \$OWNS,NOEXE,2

00000 RUNSA_ERROR:
00008 RUNSA_IMAGE:
00010 RUNSA_INPUT:
00018 RUNSA_INPUT_DESC:
00020 RUNSA_OUTPUT:
00028 RUNSA_PRCNAM:
00030 RUNSA_QUOTA:
00034 RUNSL_BASPRI:

00038 RUNSL_MBXUNT: .BLKB 4
0003C RUNSL_PID: .BLKB 4
00040 RUNSL_STATUS: .BLKB 4
00044 RUNSL_STFLG: .BLKB 4
00048 RUNSL_UIC: .BLKB 4
FFFFFFFFFF FFFFFFFF 0004C RUNSQ_DAYTIM: .LONG -1, -1
00000000 00000000 00054 RUNSQ_INTERVAL: .LONG 0, 0
0005C RUNSQ_PRVADR: .BLKB 8
00064 RUNSA_IMAGE_BUF: .BLRB 255
00163 .BLKB 1
00164 RUNSA_FINDFILE: .BLKB 4
0309 0004 00168 RUNSA_GETJPI: .WORD 4, 777
00000000' 0016C .ADDRESS RUNSL_BASPRI
00000000' 00170 .LONG 0
0204 0008 00174 .WORD 8, 516
00000000' 00178 .ADDRESS RUNSQ_PRVADR
00000000' 0017C .LONG 0
0000 0000 00180 .WORD 0, 0
00000000' 00184 .LONG 0, 0
01 0018C RUNSA_QUOTA_TBL: .BYTE 1
00000000' 0018D .ADDRESS P.ABM
03 00191 .BYTE 3
00000000' 00192 .ADDRESS P.ABO
0C 00196 .BYTE 12
00000000' 00197 .ADDRESS P.ABQ
0D 0019B .BYTE 13
00000000' 0019C .ADDRESS P.ABS
06 001A0 .BYTE 6
00000000' 001A1 .ADDRESS P.ABU
02 001A5 .BYTE 2
00000000' 001A6 .ADDRESS P.ABW
05 001AA .BYTE 5
00000000' 001AB .ADDRESS P.ABY
0A 001AF .BYTE 10
00000000' 001B0 .ADDRESS P.ACA
07 001B4 .BYTE 7
00000000' 001B5 .ADDRESS P.ACC
09 001B9 .BYTE 9
00000000' 001BA .ADDRESS P.ACE
08 001BE .BYTE 8
00000000' 001BF .ADDRESS P.ACG
04 001C3 .BYTE 4
00000000' 001C4 .ADDRESS P.AC1
0B 001C8 .BYTE 11

RUNDET
V04-000

K 13
Run Detached Process -- [LI Utility Procedure 16-Sep-1984 00:27:00 VAX-11 Bliss-32 v4.0-742
run_detached -- Create a sub or detached proces 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32:1

Page 13
(6)

00000000' 001C9 .ADDRESS P.ACK
0E 001CD :BYTE 14
00000000' 001CE .ADDRESS P.ACW

RUNS_INSVIRMEM== 12587764
RUNS_INVOQUAVAL== 12587818
RUNS_PARSEFAIL== 12587596
RUNS_SYNTAX== 12587260
ACCOUNTING= P.AAA
AUTHORIZE= P.AAC
DELAY= P.AAE
DETACH= P.AAG
DUMP= P.AAI
ERROR= P.AAK
INPUT= P.AAM
INTERVAL= P.AAO
MAILBOX= P.AAQ
OUTPUT= P.AAS
P1= P.AAU
PRIORITY= P.AAW
PRIVILEGES= P.AAY
PROCESS= P.ABA
RESOURCE_WAIT= P.ABC
SCHEDULE= P.ABE
SERVICE_FAIL= P.ABG
SWAPPING= P.ABI
UIC= P.ABK
.EXTRN CLISGET VALUE, CLISPRES
.EXTRN LIBSCVT-DTIME, LIBSCVT DX-DX
.EXTRN LIBSCVT-TIME, LIBSGET VM
.EXTRN LIBSFIND FILE, LIBSTPARSE
.EXTRN PRVSSETPRIV, CLIS_NEGATED
.EXTRN RUNS_CREPRC, RUNS_PROC_ID
.EXTRN RUNS_GETJPI, RUNS_CVTER
.EXTRN RUNS_INVUIC, RUNS_ILLVAL
.EXTRN RUNS_SCHDWK, SYSSCREPRC

.PSECT SCODES,NOWRT,2

0004 00000 RUN_DETACHED:
0000V 52 0000' CF 0000' CF 9E 00002 .WORD Save R2 0249
00 FB 00007 MOVAB RUNSL_STSFLG, R2
7E D4 0000C CALLS #0, INIT_ARG_LIST 0282
62 DD 0000E CLRL -(SP) 0295
F4 A2 DD 00010 PUSHL RUNSL_STSFLG
04 A2 DD 00013 PUSHL RUNSL_MBXUNT
F0 A2 DD 00016 PUSHL RUNSL_UIC
E4 A2 9F 00019 PUSHAB RUNSA_PRCNAM
EC A2 DD 0001C PUSHL RUNSA_QUOTA
18 A2 9F 0001F PUSHAB RUNSQ_PRVADR
BC A2 9F 00022 PUSHAB RUNSA_ERROR
DC A2 9F 00025 PUSHAB RUNSA_OUTPUT
CC A2 9F 00028 PUSHAB RUNSA_INPUT
C4 A2 9F 0002B PUSHAB RUNSA_IMAGE
F8 A2 9F 0002E PUSHAB RUNSL_PID
00 FB 00031 CALLS #13, SYSSCREPRC

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure L 13
run_detached -- Create a sub or detached process 16-Sep-1984 00:27:00 VAX-11 Bliss-32 v4.0-742
[CLIUTL.SRC]RUNDET.B32:1

Page 14
(6)

	FC A2	50 D0 00038	MOVL R0, RUNSL_STATUS	
	14	50 E8 0003C	BLBS R0, 1\$	
		FC A2 DD 0003F	PUSHL RUNSL_STATUS	0297
		7E D4 00042	CLRL -(SP)	
	00000000G 00	8F DD 00044	PUSHL #RUNS CREPRC	
		03 FB 0004A	CALLS #3, LIB\$STOP	
		12 11 00051	BRB 2\$	
		F8 A2 DD 00053 1\$:	PUSHL RUNSL_PID	0299
		01 9D 00056	PUSHL #1	
	05 00000000G 00	8F DD 00058	PUSHL #RUNS PROC_ID	
		03 FB 0005E	CALLS #3, LIB\$SIGNAL	
	0000V CF	05 E1 00065 2\$:	BBC #5, RUNSL STSFLG, 3\$	0301
		00 FB 00069	CALLS #0, SCHEDULE_PROCESS	0302
		04 0006E 3\$:	RET	0304

; Routine Size: 111 bytes. Routine Base: \$CODE\$ + 0000

: 310 0305 1

```
312 0306 1 %SBTTL 'init_arg_list -- Initialize argument lists'  
313 0307 1 ROUTINE init_arg_list : NOVALUE =  
314 0308 1 !++  
315 0309 1  
316 0310 1 Functional Description:  
317 0311 1  
318 0312 1 This routine is responsible for calling the procedures to parse  
319 0313 1 the command line and establish the arguments for creating and  
320 0314 1 scheduling the requested process.  
321 0315 1  
322 0316 1 Implicit Inputs:  
323 0317 1  
324 0318 1 Global data for the argument lists, command line entities etc..  
325 0319 1  
326 0320 1 Implicit Outputs:  
327 0321 1  
328 0322 1 The command line has been parsed with the resultant information  
329 0323 1 available for calls to the create process ($CREPRC) and the schedule  
330 0324 1 wake up ($SCHEDWK) system services.  
331 0325 1  
332 0326 1 Side Effects:  
333 0327 1  
334 0328 1 Errors encountered during initialization will be signalled by  
335 0329 1 the routine which detected the problem.  
336 0330 1  
337 0331 1 --  
338 0332 2 BEGIN  
339 0333 2  
340 0334 2 Initialize the various descriptors.  
341 0335 2  
342 0336 2 CH$FILL (0, dsc$c_s_bln, run$a_image);  
343 0337 2 CH$FILL (0, dsc$c_s_bln, run$a_input);  
344 0338 2 CH$FILL (0, dsc$c_s_bln, run$a_input_desc);  
345 0339 2 CH$FILL (0, dsc$c_s_bln, run$a_output);  
346 0340 2 CH$FILL (0, dsc$c_s_bln, run$a_error);  
347 0341 2 CH$FILL (0, dsc$c_s_bln, run$a_prcnam);  
348 0342 2 run$a_input [dsc$b_class] = dsc$k_class_d;  
349 0343 2 run$a_output [dsc$b_class] = dsc$k_class_d;  
350 0344 2 run$a_error [dsc$b_class] = dsc$k_class_d;  
351 0345 2 run$a_prcnam [dsc$b_class] = dsc$k_class_d;  
352 0346 2 run$a_input_desc [dsc$b_dtype] = dsc$k_dtype_t;  
353 0347 2 run$a_input_desc [dsc$b_class] = dsc$k_class_d;  
354 0348 2  
355 0349 3 IF NOT (run$l_status = $GETJPI (ITMLST = run$a_getjpi))  
356 0350 2 THEN SIGNAL_STOP (run$a_getjpi, 0, .run$l_status);  
357 0351 2  
358 0352 2 CLISGET_VALUE (input, run$a_input);  
359 0353 2 CLISGET_VALUE (output, run$a_output);  
360 0354 2 CLISGET_VALUE (error, run$a_error);  
361 0355 2 CLISGET_VALUE (process, run$a_prcnam);  
362 0356 2  
363 0357 2 parse_image_spec ();  
364 0358 2 get_value (mailbox, run$l_mbxit);  
365 0359 2 get_value (priority, run$l_baspri);  
366 0360 2 get_wakeup_info ();  
367 0361 2 get_uic ();  
368 0362 2 get_privileges ();
```

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure
init_arg_list -- Initialize argument lists

N 13
16-Sep-1984 00:27:00 VAX-11 Bliss-32 v4.0-742
14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32;1

Page 16
(7)

```
: 369      0363 2 get_quotas ();
: 370      0364 2 get_stsflgs ();
: 371      0365 2
: 372      0366 1 END:    ! of ROUTINE init_arg_list
:                                         ! Set up the default process quotas.
:                                         ! Set up the initial process context flags.
```

.EXTRN SYSSGETJPI

01FC 00000 INIT_ARG_LIST:

08	00	58	0000'	CF	9E	00002		WORD	Save R2,R3,R4,R5,R6,R7,R8				
08	00	57	00000000G	00	9E	00007		MOVAB	INPUT, R8				
08	00	56	0000'	CF	9E	0000E		MOVAB	CLISGET VALUE, R7				
08	00	6E	F8	00	2C	00013		MOVAB	RUNSA_INPUT, R6				
08	00	6E		66		00018		MOVCS	#0, (SP), #0, #8, RUNSA_IMAGE				
08	00	6E		08	A6	0001A		MOVCS	#0, (SP), #0, #8, RUNSA_INPUT				
08	00	6E		10	A6	0001F		MOVCS	#0, (SP), #0, #8, RUNSA_INPUT_DESC				
08	00	6E		10	A6	00025		MOVCS	#0, (SP), #0, #8, RUNSA_OUTPUT				
08	00	6E		10	A6	00027		MOVCS	#0, (SP), #0, #8, RUNSA_ERROR				
08	00	6E		10	A6	0002C		MOVCS	#0, (SP), #0, #8, RUNSA_PRCNAM				
08	00	6E		18	A6	00033		MOVCS	#0, (SP), #0, #8, RUNSA_INPUT_DESC+2				
		03	A6		02	90	0003C	MOVB	#2, RUNSA_INPUT+3				
		13	A6		02	90	00040	MOVB	#2, RUNSA_OUTPUT+3				
		F3	A6		02	90	00044	MOVB	#2, RUNSA_ERROR+3				
		1B	A6		02	90	00048	MOVB	#2, RUNSA_PRCNAM+3				
		0A	A6	020E	8F	B0	0004C	MOVW	#526, RUNSA_INPUT_DESC+2				
					7E	7C	00052	CLRQ	-(SP)				
					7E	D4	00054	CLRL	-(SP)				
				0158	C6	9F	00056	PUSHAB	RUNSA_GETJPI				
					7E	7C	0005A	CLRQ	-(SP)				
					7E	D4	0005C	CLRL	-(SP)				
		00000000G	00		07	FB	0005E	CALLS	#7, SY\$GETJPI				
		30	A6		50	DD	00065	MOVL	RO, RUNSL_STATUS				
			12		50	E8	00069	BLBS	RO, IS				
				30	A6	DD	0006C	PUSHL	RUNSL_STATUS				
					7E	D4	0006F	CLRL	-(SP)				
		00000000G	00	00000000G	8F	DD	00071	PUSHL	#RUNS_GETJPI				
					03	FB	00077	CALLS	#3, LIB\$STOP				
					56	DD	0007E	PUSHL	R6				
					58	DD	00080	PUSHL	R8				
			67		02	FB	00082	CALLS	#2, CLISGET VALUE				
					10	A6	9F	PUSHAB	RUNSA_OUTPUT				
					30	A8	9F	PUSHAB	OUTPUT				
			67		02	FB	00088	CALLS	#2, CLISGET_VALUE				
					F0	A6	9F	PUSHAB	RUNSA_ERROR				
			67		F0	A8	9F	PUSHAB	ERROR				
					02	FB	00091	CALLS	#2, CLISGET VALUE				
			67		18	A6	9F	PUSHAB	RUNSA_PRCNAM				
					74	A8	9F	PUSHAB	PROCESS				
		0000V	67		02	FB	00094	CALLS	#2, CLISGET VALUE				
					00	FB	000A0	CALLS	#0, PARSE IMAGE_SPEC				
					28	A6	9F	PUSHAB	RUNSL_MBXOUNT				

RUNDET
V04-000

Run Detached Process -- [LI Utility Procedure
init_arg_list -- Initialize argument lists

B 14

16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 v4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 17
(7)

0000V CF	20 A8 9F 000A8	PUSHAB MAILBOX	
	02 FB 000AB	CALLS #2, GET_VALUE	
	24 A6 9F 000B0	PUSHAB RUNSL_BSPRI	0359
	4C A8 9F 000B3	PUSHAB PRIORITY	
0000V CF	02 FB 000B6	CALLS #2, GET_VALUE	
0000V CF	00 FB 000BB	CALLS #0, GET_WAKEUP_INFO	0360
0000V CF	00 FB 000C0	CALLS #0, GET_UIC	0361
0000V CF	00 FB 000C5	CALLS #0, GET_PRIVILEGES	0362
0000V CF	00 FB 000CA	CALLS #0, GET_QUOTAS	0363
0000V CF	00 FB 000CF	CALLS #0, GET_STSFLGS	0364
	04 000D4	RET	0366

; Routine Size: 213 bytes, Routine Base: \$CODE\$ + 006F

: 373 0367 1

```
375      0368 1 %SBTTL 'parse_image_spec -- Parse image file specification'
376      0369 1 ROUTINE parse_image_spec : NOVALUE =
377      0370 1 !++
378      0371 1
379      0372 1 Functional Description:
380      0373 1
381      0374 1 This routine obtains the image file specification from the command
382      0375 1 line and parses it, producing a resultant file specification with
383      0376 1 defaults applied.
384      0377 1
385      0378 1 Implicit Inputs:
386      0379 1
387      0380 1     none
388      0381 1
389      0382 1 Implicit Outputs:
390      0383 1
391      0384 1 The image file has been parsed with the resultant file specification in
392      0385 1 run$sa_image_buf, and the appropriate fields of the descriptor run$sa_image
393      0386 1 initialized.
394      0387 1
395      0388 1 Side Effects:
396      0389 1
397      0390 1 Parse errors from RMS or errors from LIB$FIND_FILE will result
398      0391 1 with a FATAL error signalled.
399      0392 1
400      0393 1 --
401      0394 2 BEGIN
402      0395 2
403      0396 2 bind
404      0397 2     default_name = uplit (%charcount('.EXE'),uplit byte ('.EXE'));
405      0398 2
406      0399 2 local
407      0400 2     status
408      0401 2     findfilenam : ref block [ ,byte ];
409      0402 2
410      0403 2 CLISGET_VALUE (p1, run$sa_input_desc);           ! Obtain the image file specification.
411      0404 2
412      0405 2 run$sa_image [DSC$B_CLASS] = DSC$K_CLASS_D;
413      0406 2 run$sa_image [DSC$B_DTYPE] = DSC$K_DTYPE_T;
414      0407 2
415      0408 2 ! Get the next file name to search for, no wildcards permitted.
416      0409 2
417      0410 2 status = LIB$FIND_FILE(
418      0411 2     run$sa_input_desc, run$sa_image,
419      0412 2     run$sa_findfile
420      0413 2     default_name, 0, 0, %REF(1));
421      0414 2
422      0415 2 ! If the filename has wildcards in it it's an error
423      0416 2
424      0417 2 if (.status and sts$M_MSG_NO) eql shr$nowild
425      0418 2 then
426      0419 2     run$sa_findfile [fab$1_sts] = .status;
427      0420 2
428      0421 2 ! Report miscellaneous errors from LIB$FIND_FILE
429      0422 2
430      0423 2 if not .status
431      0424 2 then
```

```

: 632      0425 2      SIGNAL_STOP (run$_parsefail, 1,
: 633      0426      run$_input_desc, .run$_findfile [fab$1_sts], .run$_findfile [fab$1_stv]);
: 634      0427
: 635      0428      findfilenam = .run$_findfile [fab$1_nam];
: 636      0429
: 637      0430      ! If an explicit version number was not specified, remove the version number.
: 638      0431
: 639      0432      IF NOT .findfilenam [nam$v_exp_ver]
: 640      0433      THEN
: 641      0434      run$_image [dsc$w_length] = .run$_image [dsc$w_length] - .findfilenam [nam$b_ver];
: 642      0435
: 643      0436 1 END;   ! of ROUTINE parse_image_spec

```

.PSECT SPLIT\$,NOWRT,NOEXE,2

45 58 45 2E 0026C P.ACP:	.ASCII \.EXE\
00000004 00270 P.ACO:	.LONG 4
00000000 00274	.ADDRESS P.ACP

DEFAULT_NAME= P.ACO

.PSECT SCODE\$,NOWRT,2

0004 00000 PARSE_IMAGE_SPEC:

				WORD	Save R2	0369
				MOVAB	RUN\$A_FINDFILE, R2	
				SUBL2	#4, SP	0403
				PUSHAB	RUN\$A_INPUT_DESC	
				PUSHAB	P1	
				CALLS	#2, CLISGET_VALUE	0406
				MOVW	#526, RUN\$A_IMAGE+2	0413
				MOVL	#1, (SP)	
				PUSHL	SP	0410
				CLRQ	-(SP)	
				PUSHAB	DEFAULT_NAME	
				PUSHL	R2	0417
				PUSHAB	RUN\$A_IMAGE	
				PUSHAB	RUN\$A_INPUT_DESC	
				CALLS	#7, LIB\$FIND_FILE	0419
				MOVL	R0, STATUS	0423
				BICL3	#-65529, STATUS, R0	0426
				CMPL	R0, #4392	
				BNEQ	1\$	0425
				MOVL	RUN\$A_FINDFILE, R0	
				MOVL	STATUS, 8(R0)	
				BLBS	STATUS, 2\$	0428
				MOVL	RUN\$A_FINDFILE, R0	
				MOVQ	8(R0), -(SP)	
				PUSHAB	RUN\$A_INPUT_DESC	
				PUSHL	#1	
				CALLS	#12587596	
				MOVL	#5, LIB\$STOP	
				MOVL	RUN\$A_FINDFILE, R0	
					40(R0), FINDFILENAM	

RUNDET
V04-000

E 16
Run Detached Process -- [LI] Utility Procedure 16-Sep-1984 00:27:00 VAX-11 Bliss-32 V4.0-742
parse_image_spec -- Parse image file specificat 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32;1

Page 20
(8)

09	34	A0	E8	0007B	BLBS	52(FINDFILENAM), 3\$:	0432
51	3D	A0	9A	0007F	MOVZBL	61(FINDFILENAM), R1	:	0434
FEA4	C2	51	A2	00083	SUBW2	R1, RUNSA_IMAGE	:	0436
			04	00088 3\$:	RET			

; Routine Size: 137 bytes. Routine Base: \$CODE\$ + 0144

; 444 0437 1

```

446 0438 1 %SBTTL 'get_wakeup_info -- Process the $SCHEDWK time values'
447 0439 1 ROUTINE get_wakeup_info : NOVALUE =
448 0440 1 /**
449 0441 1
450 0442 1 Functional Description:
451 0443 1
452 0444 1 This routine is responsible for obtaining and converting the time values
453 0445 1 used to schedule wake up requests for the created process.
454 0446 1
455 0447 1 Implicit Outputs:
456 0448 1
457 0449 1 run$q_interval delta Reschedule the process to execute at
458 0450 1 this interval.
459 0451 1
460 0452 1 run$q_daytim absolute or Schedule the process to execute at
461 0453 1 delta this time.
462 0454 1
463 0455 1 Side Effects:
464 0456 1
465 0457 1 The time quantities are obtained as .ASCII strings from the CLI. To
466 0458 1 convert them we call the appropriate library routine. Any errors
467 0459 1 encountered during the conversion will be signalled, and execution
468 0460 1 of this image terminated.
469 0461 1
470 0462 1 --
471 0463 2 BEGIN
472 0464 2
473 0465 2 IF CLISGET_VALUE (delay, run$a_input_desc) ! Was /DELAY specified?
474 0466 2 THEN
475 0467 3 IF NOT (run$l_status = LIB$CVT_DTIME (run$a_input_desc, run$q_daytim))
476 0468 2 THEN
477 0469 2 SIGNAL_STOP (run$cvterr, 2, delay, run$a_input_desc, .run$l_status);
478 0470 2
479 0471 2 IF CLISGET_VALUE (interval, run$a_input_desc) ! Was /INTERVAL specified?
480 0472 2 THEN
481 0473 3 IF NOT (run$l_status = LIB$CVT_DTIME (run$a_input_desc, run$q_interval))
482 0474 2 THEN
483 0475 2 SIGNAL_STOP (run$cvterr, 2, interval, run$a_input_desc, .run$l_status);
484 0476 2
485 0477 2 IF CLISGET_VALUE (schedule, run$a_input_desc) ! How about /SCHEDULE?
486 0478 2 THEN
487 0479 3 IF NOT (run$l_status = LIB$CVT_TIME (run$a_input_desc, run$q_daytim))
488 0480 2 THEN
489 0481 2 SIGNAL_STOP (run$cvterr, 2, schedule, run$a_input_desc, .run$l_status);
490 0482 2
491 0483 1 END; ! of ROUTINE get_wakeup_info

```

00FC 00000 GET_WAKEUP_INFO:

57 00000000G	00	9E 00002	.WORD	Save R2,R3,R4,R5,R6,R7
56 00000000G	00	9E 00009	MOVAB	LIB\$CVT_DTIME, R7
55 00000000G	8F	00 00010	MOVAB	LIB\$STOP, R6
54 00000000G	00	9E 00017	MOVL	#RUN\$CVVERR, R5
			MOVAB	CLISGET_VALUE, R4

: 0439

RUNDET
V04-000

Run Detached Process -- [LI Utility Procedure G 14
get_wakeup_info -- Process the SSCHDWK time val 16-Sep-1984 00:27:00 VAX-11 Bliss-32 v4.0-742
14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32:1

Page 22
(9)

53	0000'	CF	9E 0001E	MOVAB	DELAY, R3	
52	0000'	CF	9E 00023	MOVAB	RUNSA_INPUT_DESC, R2	0465
		52	DD 00028	PUSHL	R2	
		53	DD 0002A	PUSHL	R3	
64		02	FB 0002C	CALLS	#2, CLISGET_VALUE	
1D		50	E9 0002F	BLBC	R0, 1\$	
	34	A2	9F 00032	PUSHAB	RUNSQ_DAYTIM	0467
		52	DD 00035	PUSHL	R2	
28	67	02	FB 00037	CALLS	#2, LIBSCVT_DTIME	
	A2	50	DO 0003A	MOVL	R0, RUNSL_STATUS	
	OE	50	E8 0003E	BLBS	R0, 1\$	
		28	A2 DD 00041	PUSHL	RUNSL_STATUS	0469
		52	DD 00044	PUSHL	R2	
		53	DD 00046	PUSHL	R3	
		02	DD 00048	PUSHL	#2	
		55	DD 0004A	PUSHL	R5	
	66	05	FB 0004C	CALLS	#5, LIB\$STOP	
		52	DD 0004F	PUSHL	R2	
		18:				
		4C	A3 9F 00051	PUSHAB	INTERVAL	0471
64	1E	02	FB 00054	CALLS	#2, CLISGET_VALUE	
		50	E9 00057	BLBC	R0, 2\$	
		3C	A2 9F 0005A	PUSHAB	RUNSQ_INTERVAL	0473
28	67	52	DD 0005D	PUSHL	R2	
	A2	02	FB 0005F	CALLS	#2, LIBSCVT_DTIME	
	OF	50	DO 00062	MOVL	R0, RUNSL_STATUS	
		50	E8 00066	BLBS	R0, 2\$	
		28	A2 DD 00069	PUSHL	RUNSL_STATUS	0473
		52	DD 0006C	PUSHL	R2	
		4C	A3 9F 0006E	PUSHAB	INTERVAL	
		02	DD 00071	PUSHL	#2	
		55	DD 00073	PUSHL	R5	
	66	05	FB 00075	CALLS	#5, LIB\$STOP	
		52	DD 00078	PUSHL	R2	
		28:				
		00D8	C3 9F 0007A	PUSHAB	SCHEDULE	0477
64	23	02	FB 0007E	CALLS	#2, CLISGET_VALUE	
		50	E9 00081	BLBC	R0, 3\$	
		34	A2 9F 00084	PUSHAB	RUNSQ_DAYTIM	0479
00000000G	00	52	DD 00087	PUSHL	R2	
28	A2	02	FB 00089	CALLS	#2, LIBSCVT_TIME	
	10	50	DO 00090	MOVL	R0, RUNSL_STATUS	
		50	E8 00094	BLBS	R0, 3\$	
		28	A2 DD 00097	PUSHL	RUNSL_STATUS	0481
		52	DD 0009A	PUSHL	R2	
		00D8	C3 9F 0009C	PUSHAB	SCHEDULE	
		02	DD 000A0	PUSHL	#2	
		55	DD 000A2	PUSHL	R5	
	66	05	FB 000A4	CALLS	#5, LIB\$STOP	
		04	000A7	RET		0483
		38:				

; Routine Size: 168 bytes, Routine Base: SCODES + 01CD

; 492 0484 1

```

: 494 0485 1 %SBTTL 'get_uic -- Process the UIC, converting it to a longword value'
: 495 0486 1 ROUTINE get_uic : NOVALUE =
: 496 0487 1 ++
: 497 0488 1 Functional Description:
: 498 0489 1 This routine is responsible for obtaining and converting a UIC string
: 499 0490 1 of the form [group, member] to a longword value.
: 500 0491 1 Implicit Inputs:
: 501 0492 1 run$u_input_desc adr Address of a general purpose dynamic
: 502 0493 1 string descriptor.
: 503 0494 1 Side Effects:
: 504 0495 1 If the UIC could not be converted, we will inform the user with
: 505 0496 1 an invalid UIC diagnostic and exit.
: 506 0497 1 --
: 507 0498 1 BEGIN
: 508 0499 1 LOCAL
: 509 0500 1 TPA_PARAM : $BBLOCK [TPASK_LENGTH0]
: 510 0501 1 INITIAL (REP TPASK_LENGTH0 OF BYTE (0));
: 511 0502 2 IF CLISGET_VALUE (uic, run$u_input_desc)           ! If the user supplied a UIC
: 512 0503 2 THEN                                         ! convert it to a longword value.
: 513 0504 3 BEGIN
: 514 0505 3 tpa_param [tpa$1_count] = tpask_count0;
: 515 0506 3 tpa_param [tpa$1_stringcnt] = .run$u_input_desc [dsc$w_length];
: 516 0507 3 tpa_param [tpa$1_stringptr] = .run$u_input_desc [dsc$sa_pointer];
: 517 0508 3 IF NOT LIBSTPARSE (tpa_param, uic_states, uic_keys)
: 518 0509 3 THEN
: 519 0510 3 SIGNAL_STOP (run$invuic, 1, run$u_input_desc); ! converting the UIC, inform the user.
: 520 0511 3 END;
: 521 0512 2 END;      ! of ROUTINE get_uic

```

.PSECT \$SPLIT\$,NOWRT,NOEXE,2
 00# 00278 P.ACQ: .BYTE 0[36]

							.PSECT \$CODES,NOWRT,2	
6E	0000'	CF	56	0000'	007C 000000 GET_UIC:.WORD	MOVAB	Save R2,R3,R4,R5,R6	: 0486
			5E		9E 00002	SUBL2	RUNSA INPUT_DESC, R6	
					24 C2 00007	SP	#36,	
					24 28 0000A	MOVC3	#36, P.ACQ, TPA_PARAM	: 0509
					56 DD 00010	PUSHL	R6	: 0511
					CF 9F 00012	PUSHAB	UIC	
	0000000G	00		0000'	02 FB 00016	CALLS	#2, CLISGET_VALUE	

RUNDET
V04-000

I 14
Run Detached Process -- CLI Utility Procedure 16-Sep-1984 00:27:00 VAX-11 Bliss-32 v4.0-742
get_uic -- Process the UIC, converting it to a 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32;1

Page 24
(10)

32	50	E9 0001D	BLBC	R0, 1\$	
6E	08	D0 00020	MOVL	#8, TPA PARAM	0514
08 AE	66	3C 00023	MOVZWL	RUNSA_INPUT_DESC, TPA_PARAM+8	0515
0C AE	04 A6	D0 00027	MOVL	RUNSA_INPUT_DESC+4, TPA_PARAM+12	0516
	0000' CF	9F 0002C	PUSHAB	UIC KEYS	0518
	0000' CF	9F 00030	PUSHAB	UIC STATES	
	08 AE	9F 00034	PUSHAB	TPA_PARAM	
00000000G 00	03 FB	00037	CALLS	#3, LIB\$PARSE	
	11	50 E8 0003E	BLBS	R0, 1\$	
		56 DD 00041	PUSHL	R6	0520
		01 DD 00043	PUSHL	#1	
00000000G 00	00000000G 8F	DD 00045	PUSHL	#RUNS INVUIC	
		03 FB 0004B	CALLS	#3, LIB\$STOP	
		04 00052 1\$:	RET		0523

: Routine Size: 83 bytes. Routine Base: \$CODE\$ + 0275

: 533 0524 1

```

535 1 %SBTTL 'get_privileges -- Obtain the process privileges
536 1 ROUTINE get_privileges : NOVALUE =
537 1 !++
538
539 1 Functional Description:
540
541 1 This routine iteratively calls the routine CLISGET_VALUE to obtain the
542 1 privileges specified by the user. The private CLI routine PRVSSETPRIV
543 1 is then called to convert the ascii string name into a bit number and
544 1 set/clear the appropriate bit in the privilege mask.
545
546 1 Implicit Inputs:
547
548 0538 1 prv$sa_input_desc adr Address of a general purpose dynamic
549 0539 1 string descriptor.
550 0540 1 prv$q_prvadr adr Address of the privilege mask quadword.
551
552 0542 1 Implicit Outputs:
553
554 0544 1 The privilege vector has been established. The privileges are set/cleared
555 1 according to whether they were explicitly specified, explicitly negated
556 1 or set as a result of the 'SAME' privilege.
557
558 0548 1 Side Effects:
559
560 0550 1 The 'SAME' privilege is special cased...
561
562 0552 1 If an invalid privilege keyword is detected we will signal a fatal error.
563
564 0554 1 !--
565 0555 2 BEGIN
566
567 0556 2 BIND
568 0557 2 same = $descriptor ('SAME'); $block,
569 0558 2 nosame = $descriptor ('NOSAME'); $block;
570
571 0560 2 IF (CLISPRESENT ($descriptor ('PRIVILEGES,SAME')) EQL clis_negated)           ! Default action is to use t
572 0561 2 THEN run$q_prvadr [1] = run$q_prvadr [0] = 0;                                ! privileges as the creating
573
574 0563 2 WHILE CLISGET_VALUE (privileges, run$sa_input_desc)                         ! Preclude an invalid keyword
575 0564 2 DO IF CHSNEQ (.run$sa_input_desc [dsc$w_length], .run$sa_input_desc [dsc$sa_pointer], .same [dsc$w_length], .s
576 0565 2 AND CHSNEQ (.run$sa_input_desc [dsc$w_length], .run$sa_input_desc [dsc$sa_pointer], .nosame [dsc$w_length], .n
577 0566 2 THEN IF NOT prv$setpriv (run$sa_input_desc, run$q_prvadr)                      ! screening out the '[NO]SAM'
578 0567 2 THEN SIGNAL_STOP (run$invquaval, 2, run$sa_input_desc, privileges);          ! [NO]SAM
579
580 0568 2
581
582 0571 2 END: ! of ROUTINE get_privileges

```

.PSECT SPLITS,NOWRT,NOEXE,2

45 4D 41 53 0029C P.ACS:	.ASCII \SAME\
00000004 002A0 P.ACR:	.LONG 4
00000000 002A4	.ADDRESS P.ACS
45 4D 41 53 4F 4E 002A8 P.ACU:	.ASCII \NOSAME\

RUNDET
V04-000

K 14
Run Detached Process -- CLI Utility Procedure 16-Sep-1984 00:27:00 VAX-11 Bliss-32 V4.0-742
get_privileges -- Obtain the process privileges 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32:1

Page 26
(11)

45 4D 41 53 2E 53 45 47 45 4C 49 56 49 52 50 00000006 002AE .BLKB 2
00000000 002B0 P.ACT: .LONG 6
00000000 002B4 .ADDRESS P.ACW
002B8 P.ACW: .ASCII \PRIVILEGES.SAME\
002C7 .BLKB 1
0000000F 002C8 P.ACV: .LONG 15
00000000 002CC .ADDRESS P.ACW
;
;
;
;

SAME= P.ACW
NOSAME= P.ACT

.PSECT \$CODES,NOWRT,2

001C 00000 GET_PRIVILEGES:
0000' CF 54 0000' CF 9E 00002 .WORD Save R2,R3,R4 : 0526
00000000G 00 0000' CF 9F 00007 MOVAB RUNSA_INPUT_DESC, R4
00000000G 8F 01 FB 0000B PUSHAB P.ACV : 0561
00000000G 00 0000' 01 FB 0000B CALLS #1, CLISPRESENT
00000000G 40 50 D1 00012 CMPL R0, #CLIS_NEGATED
00000000G 00 0000' 03 12 00019 BNEQ 1\$: 0562
00000000G 00 0000' A4 7C 0001B CLRQ RUNSQ_PRVADR
00000000G 40 54 DD 0001E 1\$: PUSHAB R4 : 0564
00000000G 00 0000' CF 9F 00020 PRIVILEGES
00000000G 00 0000' 02 FB 00024 CALLS #2, CLISGET_VALUE
00000000G 40 50 E9 0002B BLBC R0, 2\$: 0566
0000' CF 0000' 64 2D 0002E CMPCS RUNSA_INPUT_DESC, @RUNSA_INPUT_DESC+4, #0, - SAME, @SAME+4
0000' CF 00 04 B4 0000' DF 00036 BEQL : 0567
0000' CF 00 04 B4 0000' E3 13 00039 CMPCS RUNSA_INPUT_DESC, @RUNSA_INPUT_DESC+4, #0, - NOSAME, @NOSAME+4
00000000G 00 C7 0000' 64 2D 0003B BEQL 1\$: 0569
00000000G 00 0000' DF 00043 PUSHAB RUNSQ_PRVADR
00000000G 00 0000' D6 13 00046 PUSHAB R4 : 0570
00000000G 00 0000' A4 9F 00048 PRIVILEGES
00000000G 00 0000' 54 DD 0004B PUSHAB #2, PRV\$SETPRIV
00000000G 00 C7 0000' 02 FB 0004D CALLS R0, 1\$
00000000G 00 0000' 50 E8 00054 BLBS PUSHAB PRIVILEGES
00000000G 00 0000' CF 9F 00057 PUSHAB R4
00000000G 00 0000' 54 DD 0005B PUSHAB #2
00000000G 00 0000' 02 DD 0005D PUSHAB #1258781?
00000000G 00 0000' 8F DD 0005F CALLS #4, LIB\$STOP
00000000G 00 0000' 04 FB 00065 BRB 1\$: 0566
00000000G 00 0000' 80 11 0006C RET : 0572
00000000G 00 0000' 04 0006E 2\$: ;
;

: Routine Size: 111 bytes. Routine Base: \$CODES + 02C8

: 583 0573 1

```
585      0574 1 %SBTTL 'get_quotes -- Obtain the process quota values'  
586      0575 1 ROUTINE get_quotes : NOVALUE =  
587      0576 1 ++  
588  
589      0577 1 Functional Description:  
590  
591      0578 1 This routine is responsible for establishing the quota list.  
592  
593      0582 1 NOTE: The ending address of the quota table computation in the INCR  
594          0583 1 loop below subtracts 2 from the pql$_length value. This is the  
595          0584 1 correction value for the unused pql$_listend entry and the true  
596          0585 1 ending address of the quota table.  
597  
598      0587 1 Implicit Inputs:  
599  
600      0589 1 The quota list table.  
601  
602      0590 1 Quota table format:  
603  
604      0593 1      7      0  
605          +-----+  
606          ! name !  
607          +-----+-----+  
608          ! cmnd line ent desc !  
609          +-----+-----+  
610  
611      0600 1 Where the name field contains the PQLS_xxx value  
612          0601 1 and the cmnd line ent desc contains the address  
613          0602 1 of the quota name descriptor.  
614  
615      0604 1 Implicit Outputs:  
616  
617      0606 1 The quota list is established.  
618  
619      0608 1 --  
620      0609 2 BEGIN  
621  
622      0611 2 LOCAL  
623          0612 2     value : volatile;                                ! Command line entity's value.  
624  
625      0613 2     INCR entry FROM run$a_quota_tbl  
626          0615 3           TO run$a_quota_tbl + ((pql$_length - 2) * list_k_entry_size) ! Search through the quota list looking for  
627          0616 2           BY list_k_entry_size  
628  
629      0617 2     DO  
630          0618 3       BEGIN  
631          0619 3       BIND quota_entry = .entry : $bblock;  
632          0620 3       IF CLISPRESNT (.quota_entry [list_l_value])  
633          0621 3       THEN  
634          0622 4         BEGIN  
635          0623 4         IF .quota_entry [list_b_name] EQL pql$_cpulm  
636          0624 4         THEN   get_cpulm (.quota_entry [list_l_value], value)  
637          0625 4         ELSE   get_value (.quota_entry [list_l_value], value);  
638          0626 4         insert_quota (.quota_entry [list_b_name], .value);  
639          0627 3         END;  
640          0628 2       END;    ! of DO statement.  
641          0629 2  
642          0630 2     IF .run$a_quota NEQ 0  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

RUNDET
V04-000

Rur Detached Process -- CLI Utility Procedure
get_quotas -- Obtain the process quota values

M 14
16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 28
(12)

: 642 0631 2 THEN insert_quota (pqls_listend, 0);
: 643 0632 2
: 644 0633 1 END; ! of ROUTINE get_quotas

! terminate the list.

000C 00000 GET_QUOTAS:
.WORD Save R2,R3
52 0000' CF 9E 00002 MOVAB RUNSA_QUOTA_TBL, R2
53 0000' CF 9E 00007 MOVAB RUNSA_QUOTA_TBL+65, R3
35 10 0000C BSSB \$S
01 A2 DD 0000E 1\$: PUSHL 1(ENTRY)
00000000G 00 01 FB 00011 CALLS #1, CLISPRES
25 01 50 E9 00018 BLBC R0, 4\$
04 01 62 91 0001B CMPB (ENTRY), #4
0C 12 0001E BNEQ 2\$
5E DD 00020 PUSHL SP
0000V CF 01 A2 DD 00022 PUSHL 1(ENTRY)
02 FB 00025 CALLS #2, GET_CPLM
0A 11 0002A BRB 3\$
0000V CF 01 5E DD 0002C 2\$: PUSHL SP
02 FB 00031 CALLS 1(ENTRY)
0000V CF 01 A2 DD 0002E PUSHL #2, GET_VALUE
6E DD 00036 3\$: PUSHL VALUE
7E 02 FB 0003B MOVZBL (ENTRY), -(SP)
0000V CF 02 62 9A 00038 CALLS #2, INSERT_QUOTA
52 05 C0 00040 4\$: ADDL2 #5, ENTRY
53 05 52 D1 00043 5\$: CMPL ENTRY, R3
0000' CF C6 15 00046 BLEQ 1\$
07 07 D5 00048 TSTL RUNSA_QUOTA
0000V CF 02 13 0004C BEQL 6\$
0000V CF 02 7E 7C 0004E CLRQ -(SP)
04 00055 6\$: CALLS #2, INSERT_QUOTA
RET ; 0633

: Routine Size: 86 bytes, Routine Base: SCODE\$ + 0337

: 645 0634 1

```
647 0635 1 %SBTTL 'get_cpulm -- Special case the CPU time limit quota'
648 0636 1 ROUTINE get_cpulm (entry, value) : NOVALUE =
649 0637 1 ++
650 0638 1
651 0639 1 Functional Description:
652 0640 1
653 0641 1 This routine handles the special case of the CPU time limit quota,
654 0642 1 since a simple string to numeric conversion is not the case.
655 0643 1
656 0644 1 We perform the CLI call back locally to get the cpu time limit string,
657 0645 1 then we call the time conversion routine followed by reducing the time
658 0646 1 value to a single longword.
659 0647 1
660 0648 1 Inputs:
661 0649 1
662 0650 1 entry adr Address of the string descriptor for the
663 0651 1 cpu time limit quota command qualifier.
664 0652 1
665 0653 1 Outputs:
666 0654 1
667 0655 1 value adr The address of the resultant time value.
668 0656 1
669 0657 1 Side Effects:
670 0658 1
671 0659 1 If any errors are encountered the will be signalled.
672 0660 1
673 0661 1 --
674 0662 2 BEGIN
675 0663 2
676 0664 2 BUILTIN
677 0665 2 EDIV;
678 0666 2
679 0667 2 LOCAL
680 0668 2 delta : VECTOR [2, LONG];
681 0669 2 psl : $bblock [4];                                ! Delta time quadword.
682 0670 2
683 0671 2 MAP
684 0672 2 entry : REF $bblock;
685 0673 2 value : REF $bblock;
686 0674 2
687 0675 2 CLISGET VALUE (.entry, run$sa_input_desc);
688 0676 3 IF (run$1_status = LIB$CVT_DTIME (run$sa_input_desc, delta)) ! First we must get the time string and
689 0677 2 THEN convert the .ASCII string to a delta time.
690 0678 3 BEGIN If conversion was ok then condense the del
691 0679 3 psl = EDIV (%REF(-200000), delta [0], delta [0], delta [1]); time to a single longword value.
692 0680 3 Convert the delta time to a CPU time limit
693 0681 3 IF .psl [psl$v_v]
694 0682 3 THEN SIGNAL_STOP (run$_illval, 1, run$sa_input_desc); ! Check for overflow.
695 0683 3 ! CPU time limit value was too big.
696 0684 3 IF .delta [1] NEQ 0
697 0685 3 THEN delta [1] = 1;                                ! No overflow...should we round the
698 0686 3 ! CPU time limit?
699 0687 3 .value = (.delta [0] * 2) + .delta [1];           ! Set the CPU time limit.
700 0688 3 END
701 0689 2 ELSE SIGNAL_STOP (run$_syntax, 1, run$sa_input_desc, .run$1_status);
702 0690 2
703 0691 2
```

: 704 0692 1 END; ! of ROUTINE get_cpulm

B 15

000C 00000 GET_CPULM:									
							.WORD	Save R2,R3	0636
			53 0000000G	00 9E 00002			MOVAB	LIB\$STOP, R3	
			52 0000'	CF 9E 00009			MOVAB	RUNSA_INPUT_DESC, R2	
			5E	08 C2 0000E			SUBL2	#8, SP	
				52 DD 00011			PUSHL	R2	
				04 AC DD 00013			PUSHL	ENTRY	0675
			0000000G	00 02 FB 00016			CALLS	#2, CLISGET_VALUE	
				4004 8F BB 0001D			PUSHR	#^M<R2,SP>	0676
			0000000G	00 02 FB 00021			CALLS	#2, LIB\$CVT_DTIME	
			28 A2	50 D0 00028			MOVL	R0, RUNSL_STATUS	
				30 50 E9 0002C			BLBC	R0, 3S	
04 AE	6E	6E FFFCF2C0		8F 7B 0002F			EDIV	#-200000, DELTA, DELTA, DELTA+4	0679
	0D	50		50 DC 00039			MOVPSL	R0	
				01 E1 0003B			BBC	#1, PSL, 1\$	0681
				52 DD 0003F			PUSHL	R2	0682
				01 DD 00041			PUSHL	#1	
			63 0000000G	8F DD 00043			PUSHL	#RUNS_ILLVAL	
				03 FB 00049			CALLS	#3, LIB\$STOP	
				04 AE D5 0004C	1\$:		TSTL	DELTA+4	0684
				04 13 0004F			BEQL	2S	
04 AE				01 DD 00051			MOVL	#1, DELTA+4	0685
				50 6E DD 00055	2\$:		MOVL	DELTA, R0	0687
08 BC	04 BE40			04 3E 00058			MOVAW	@DELTA+4[R0], @VALUE	
				04 0005E			RET		0676
				28 A2 DD 0005F	3\$:		PUSHL	RUNSL_STATUS	0690
				52 DD 00062			PUSHL	R2	
				01 DD 00064			PUSHL	#1	
			63 00C010FC	8F DD 00066			PUSHL	#12587260	
				04 FB 0006C			CALLS	#4, LIB\$STOP	
				04 0006F			RET		0692

: Routine Size: 112 bytes. Routine Base: \$CODE\$ + 038D

: 705 0693 1

```

707 0694 1 %SBTTL 'get_value -- Obtain and convert a command line entity'
708 0695 1 ROUTINE get_value (entry, ret_val) : NOVALUE =
709 0696 1 ++
710 0697 1 Functional Description:
711 0698 1
712 0699 1 This routine will obtain a command line entity and convert
713 0700 1 the text representation of the value into a numeric quantity.
714 0701 1
715 0702 1 Inputs:
716 0703 1
717 0704 1 entry adr The address of the current command line quota entity
718 0705 1 in the quota table.
719 0706 1
720 0707 1 Outputs:
721 0708 1
722 0709 1 ret_val adr The address of a longword to receive the converted value
723 0710 1 of the command line entity.
724 0711 1
725 0712 1 Side Effects:
726 0713 1
727 0714 1 Errors encountered during conversion will be signalled.
728 0715 1
729 0716 1
730 0717 1 --
731 0718 2 BEGIN
732 0719 2
733 0720 2 LOCAL
734 0721 2 value_desc : $bblock [dsc$e_s_bln], ! Descriptor for conversion to a numeric val
735 0722 2 value; ! Resultant value from conversion.
736 0723 2
737 0724 2 MAP
738 0725 2 entry : REF $bblock,
739 0726 2 ret_val : REF $bblock;
740 0727 2
741 0728 2 value_desc [dsc$w_length] = 0; ! Length is left upto the conversion routine
742 0729 2 value_desc [dsc$b_dtype] = dsc$k_dtype_lu; ! Type is unsigned longword.
743 0730 2 value_desc [dsc$b_class] = dsc$k_class_s; ! Scalar.
744 0731 2 value_desc [dsc$a_pointer] = value; ! Address to store result of conversion.
745 0732 2
746 0733 2 IF CLISGET_VALUE (.entry, run$e_input_desc) ! If the entity is present in the command
747 0734 2 THEN ! line and has a value associated with it,
748 0735 3 IF NOT (run$! status = ! perform the conversion according to the in
749 0736 3 LIB$CVT_DX_DX (run$e_input_desc, value_desc)) ! supplied by the descriptors.
750 0737 2 THEN SIGNAL_STOP (run$cvterr, 2, .entry, run$e_input_desc, .run$! status)
751 0738 2 ELSE .ret_val = .value;
752 0739 2
753 0740 1 END: ! of ROUTINE get_value
INFO#250 L1:0738
; Referenced LOCAL symbol VALUE is probably not initialized

```

0004 00000 GET_VALUE:
 S2 0000' CF 9E 00002 :WORD Save R2
 MOVAB RUN\$A_INPUT_DESC, R2

: 0695

RUNDET
V04-000

D 15
Run Detached Process -- CLI Utility Procedure 16-Sep-1984 00:27:00 VAX-11 Bliss-32 V4.0-742
get_value -- Obtain and convert a command line 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32;1

Page 32
(14)

04	SE	01040000	0C	C2	00007	SUBL2	#12 SP	: 0728
08	AE		8F	DD	0000A	MOVL	#17039360, VALUE_DESC	: 0731
			6E	9E	00012	MOVAB	VALUE, VALUE_DESC+4	: 0733
			52	DD	00016	PUSHL	R2	
00000000G	00		04	AC	DD 00018	PUSHL	ENTRY	
	2F		02	FB	0001B	CALLS	#2, CLISGET_VALUE	
			50	E9	00022	BLBC	R0, 2\$	
			04	AE	9F 00025	PUSHAB	VALUE_DESC	: 0736
00000000G	00		52	DD	00028	PUSHL	R2	
	28		02	FB	0002A	CALLS	#2, LIB\$CVT DX DX	
	A2		50	DD	00031	MOVL	R0, RUNSL_STATUS	
	18		50	E8	00035	BLBS	R0, 18	
			28	A2	DD 00038	PUSHL	RUNSL_STATUS	: 0737
			52	DD	0003B	PUSHL	R2	
			04	AC	DD 0003D	PUSHL	ENTRY	
00000000G	00	00000000G	02	DD	00040	PUSHL	#2	
			05	FB	00042	PUSHL	#RUNS CVTERR	
			04	0004F		CALLS	#5, LIB\$STOP	
	08	BC	6E	DD	00050 18:	RET		
			04	00054	28:	MOVL	VALUE, RET_VAL	: 0738
						RET		: 0740

; Routine Size: 85 bytes. Routine Base: SCODE\$ + 03FD

; 754 0741 1

```
756 0742 1 %SBTTL 'insert_quota -- Insert a quota into the quota list'  
757 0743 1 ROUTINE insert_quota (name, value) : NOVALUE =  
758 0744 1 !++  
759 0745 1 Functional Description:  
760 0746 1 This routine is responsible for entering the specified  
761 0747 1 quota list entry into the quota list.  
762 0748 1 Inputs:  
763 0749 1 name val The name of the quota (pqls_xxx)  
764 0750 1 value val The value of the quota.  
765 0751 1 Implicit Outputs:  
766 0752 1 The quota entry has been added to the quota list and  
767 0753 1 the list pointer has been updated to point to the next  
768 0754 1 entry slot.  
769 0755 1 Side Effects:  
770 0756 1 If this is the first call, we will allocate a heap of  
771 0757 1 memory to store the quota list in. This will be  
772 0758 1 deallocated at image rundown.  
773 0759 1 --  
774 0760 2 BEGIN  
775 0761 2 OWN  
776 0762 2 quota_ptr : REF Sblock [4];  
777 0763 2 IF .run$a_quota EQL 0  
778 0764 2 THEN  
779 0765 3 BEGIN  
780 0766 4 IF NOT (run$1_status =  
781 0767 4 LIB$GET_VM (%REF(list_k_entry_size * pqls_length),  
782 0768 4 run$a_quota))  
783 0769 3 THEN  
784 0770 3 SIGNAL_STOP (run$_insvirmem, 0, .run$1_status);  
785 0771 3 quota_ptr = .run$a_quota;  
786 0772 END;  
787 0773 2 quota_ptr [list_b_name] = .name;  
788 0774 2 quota_ptr [list_l_value] = .value;  
789 0775 2 quota_ptr = .quota_ptr + list_k_entry_size;  
790 0776 2 END; ! of ROUTINE insert_quota
```

.PSECT \$0WNS,NOEXE,2

001D2 .BLKB 2
001D4 QUOTA_PTR:

.BLKB 4

.PSECT \$CODES,NOWRT,2

0004 00000 INSERT_QUOTA:

				WORD	Save R2	: 0743
				MOVAB	RUNSA_QUOTA, R2	
				SUBL2	#4, SP	: 0775
				TSTL	RUNSA_QUOTA	
				BNEQ	2\$	
				PUSHL	R2	: 0779
				MOVZBL	#75, 4(SP)	
				PUSHAB	4(SP)	
				CALLS	#2, LIB\$GET_VM	
				MOVL	R0, RUNSL_STATUS	
				BLBS	R0, 1\$	
				PUSHL	RUNSL_STATUS	: 0782
				CLRL	-(SP)	
				PUSHL	#12587764	
				CALLS	#3, LIB\$STOP	
				MOVL	RUNSA_QUOTA, QUOTA_PTR	: 0784
				MOVL	QUOTA_PTR, R0	: 0787
				MOVB	NAME_(R0)	
				MOVL	VALUE_1(R0)	: 0788
				ADDL2	#5, QUOTA_PTR	: 0789
				RET		: 0791

: Routine Size: 81 bytes. Routine Base: \$CODES + 0452

: 806 0792 1

```

808 0793 1 %SBTTL 'get_stsflgs -- Set up initial process state flags'
809 0794 1 ROUTINE get_stsflgs : NOVALUE =
810 0795 1 ++
811 0796 1
812 0797 1 Functional Description:
813 0798 1
814 0799 1 This routine performs call backs to the CLI to obtain the initial
815 0800 1 process state flag settings.
816 0801 1
817 0802 1 Implicit Inputs:
818 0803 1
819 0804 1 runSL_stsflg      adr      The address of the status flags vector.
820 0805 1
821 0806 1 Implicit Outputs:
822 0807 1
823 0808 1 runSL_stsflg      adr      The various state flags have been set.
824 0809 1
825 0810 1 Side Effects:
826 0811 1
827 0812 1 Several state flags have been defaulted to false. They are:
828 0813 1 BATCH, INTER, DISAWS and NETWRK.
829 0814 1
830 0815 1 --
831 0816 2 BEGIN
832 0817 2
833 0818 2 runSL_stsflg [prc$V_ssrrwait] = NOT (CLISPRESENT (resource_wait)); ! Resource wait mode.
834 0819 2 runSL_stsflg [prc$V_ssrexcu] = CLISPRESENT (service_fail); System service failure exception mode.
835 0820 2 runSL_stsflg [prc$V_pswapm] = NOT (CLISPRESENT (swapping)); Process swap mode.
836 0821 2 runSL_stsflg [prc$V_noacnt] = NOT (CLISPRESENT (accounting)); Process accounting.
837 0822 2 runSL_stsflg [prc$V_batch] = false; Not a batch process.
838 0823 2 runSL_stsflg [prc$V_inter] = false; Not an interactive process.
839 0824 2 runSL_stsflg [prc$V_hiber] = CLISPRESENT (delay) OR Process hibernation state can be set
840 0825 2                                CLISPRESENT (interval) OR by any combination of the /DELAY,
841 0826 2                                CLISPRESENT (schedule); /INTERVAL or /SCHEDULE command qualifiers.
842 0827 2 runSL_stsflg [prc$V_login] = NOT (CLISPRESENT (authorize)); Process authorization if image is LOGINOUT
843 0828 2 runSL_stsflg [prc$V_netwrk] = false; Not a network connect object.
844 0829 2 runSL_stsflg [prc$V_disaws] = false; Automatic working set adjustment state.
845 0830 2 runSL_stsflg [prc$V_imgdmp] = CLISPRESENT (dump); Dump requested
846 0831 2 runSL_stsflg [prc$V_detach] = CLISPRESENT (detach); Detached process
847 0832 2
848 0833 1 END:      ! of ROUTINE get_stsflgs

```

007C 00000 GET_STSFLGS:						
				.WORD		0794
56	0000'	CF	9E 00002	MOVAB	Save R2, R3, R4, R5, R6	
55	0000'	CF	9E 00007	MOVAB	RESOURCE_WAIT, R6	
54	00000000G	00	9E 0000C	MOVAB	RUNSL_STSFLG, RS	
		56	DD 00013	PUSHL	CLISPRESENT, R4	
		64	01 FB 00015	CALLS	R6	
		51	50 D2 00018	MCOML	#1, CLISPRESENT	0818
		01	51 F0 0001B	INSV	R0, R1	
		28	A6 9F 00020	PUSHAB	R1, #0, #1, RUNSL_STSFLG	
		64	01 FB 00023	CALLS	SERVICE FAIL	0819
					#1, CLISPRESENT	

RUNDET
V04-000

Run Detached Process -- [LI Utility Procedure H 15
get_stsflgs -- Set up initial process state fla 16-Sep-1984 00:27:00 VAX-11 Bliss-32 v4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 36
(16)

65	01	01	50	F0 00026	INSV	R0, #1, #1, RUNSL_STSFLG	:	0820
		38	A6 9F 00028	PUSHAB	SWAPPING			
		64	01 FB 0002E	CALLS	#1, CLISPRES			
		51	50 D2 00031	MCOML	R0, R1			
65	01	02	51 F0 00034	INSV	R1, #2, #1, RUNSL_STSFLG	:	0821	
		FF14	C6 9F 00039	PUSHAB	ACCOUNTING			
		64	01 FB 0003D	CALLS	#1, CLISPRES			
		51	50 D2 00040	MCOML	R0, R1			
65	01	03	51 F0 00043	INSV	R1, #3, #1, RUNSL_STSFLG	:	0823	
		65	0410 8F AA 00048	BICW2	#1040, RUNSL_STSFLG		0824	
		FF38	C6 9F 0004D	PUSHAB	DELAY			
		64	01 FB 00051	CALLS	#1, CLISPRES			
		53	50 D0 00054	MOVL	R0, R3			
		84	A6 9F 00057	PUSHAB	INTERVAL	:	0825	
		64	01 FB 0005A	CALLS	#1, CLISPRES			
		52	50 D0 0005D	MOVL	R0, R2			
		52	53 C8 00060	BISL2	R3, R2			
		10	A6 9F 00063	PUSHAB	SCHEDULE	:	0826	
		64	01 FB 00066	CALLS	#1, CLISPRES			
65	53	50	52 89 00069	BISB3	R2, R0, R3			
	01	05	53 F0 0006D	INSV	R3, #5, #1, RUNSL_STSFLG	:	0827	
		FF28	C6 9F 00072	PUSHAB	AUTHORIZE			
		64	01 FB 00076	CALLS	#1, CLISPRES			
		51	50 D2 00079	MCOML	R0, R1			
65	01	06	51 F0 0007C	INSV	R1, #6, #1, RUNSL_STSFLG	:	0829	
		65	0180 8F AA 00081	BICW2	#384, RUNSL_STSFLG		0830	
		FF54	C6 9F 00086	PUSHAB	DUMP			
01 AS	01	64	01 FB 0008A	CALLS	#1, CLISPRES			
		03	50 F0 0008D	INSV	R0, #3, #1, RUNSL_STSFLG+1	:	0831	
01 AS	01	64	FF48 C6 9F 00093	PUSHAB	DETACH			
		01	01 FB 00097	CALLS	#1, CLISPRES			
		50	F0 0009A	INSV	R0, #1, #1, RUNSL_STSFLG+1			
		04 000A0	RET			:	0833	

: Routine Size: 161 bytes, Routine Base: SCODES + 04A3

: 849 0834 1

```

: 851    0835 1 %SBTTL 'schedule_process -- Schedule the process for execution'
: 852    0836 1 ROUTINE schedule_process : NOVALUE =
: 853    0837 1 !++
: 854    0838 1
: 855    0839 1 Functional Description:
: 856    0840 1
: 857    0841 1 This routine is responsible for scheduling wake up request(s) for the
: 858    0842 1 created process.
: 859    0843 1
: 860    0844 1 Implicit Inputs:
: 861    0845 1
: 862    0846 1 run$1_pid      adr   Address of the created process' PID.
: 863    0847 1 run$1_daytim   adr   Address of the wake up time quadword.
: 864    0848 1 run$1_interval adr   Address of the repeat time quadword.
: 865    0849 1
: 866    0850 1 Implicit Outputs:
: 867    0851 1
: 868    0852 1 The wake up request has been scheduled.
: 869    0853 1
: 870    0854 1 Side Effects:
: 871    0855 1
: 872    0856 1 If any error is encountered while attempting to schedule the wake up
: 873    0857 1 request(s), we will signal a warning diagnostic to the user and exit.
: 874    0858 1
: 875    0859 1 --
: 876    0860 2 BEGIN
: 877    0861 2
: 878    P 0862 3 IF NOT (run$1_status = $SCHDWK (PIDADR = run$1_pid,
: 879    P 0863 3                      DAYTIM = run$1_daytim,
: 880    P 0864 3                      REPTIM = run$1_interval))      ! Schedule a wake up for the created process
: 881    P 0865 2 THEN          ! At the specified time of day.
: 882    P 0866 2 SIGNAL (run$1_schdwk, 1, .run$1_pid, .run$1_status);  ! Repeat the scheduled wake up at this inter
: 883    P 0867 2
: 884    0868 1 END;      ! of ROUTINE schedule_process      ! Report any problems scheduling the wake up

```

.EXTRN SYSSCHDWK

0004 00000 SCHEDULE_PROCESS:

52	0000'	CF 9E 00002	.WORD Save R2	: 0836
		18 A2 9F 00007	MOVAB RUNSL_PID, R2	: 0864
		10 A2 9F 0000A	PUSHAB RUNSQ_INTERVAL	: 0864
		7E D4 0000D	PUSHAB RUNSQ_DAYTIM	: 0864
		52 DD 0000F	CLRL -(SP)	: 0864
00000000G 00	04	FB 00011	PUSHL R2	: 0864
04 A2	50	D0 00018	CALLS #4, SYSSCHDWK	: 0864
12	50	F8 0001C	MOVL R0, RUNSL_STATUS	: 0864
7E	62	7D 0001F	BLBS R0, 1\$: 0864
		01 DD 00022	MOVO RUNSL_PID, -(SP)	: 0866
00000000G 00	8F	DD 00024	PUSHL #1	: 0866
		04 FB 0002A	CALLS #4, LIB\$SIGNAL	: 0868
		04 00031 1\$:	RET	: 0868

; Routine Size: 50 bytes, Routine Base: SCODES + 0544

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure J 15
schedule_process -- Schedule the process for ex 16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 v4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 38
(17)

: 885 0869 1
: 886 0870 1 END ! of MODULE rundet
: 887 0871 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
SPLITS	720	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
SOWNS	472	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
-LIB\$KEYOS	0	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1)
-LIB\$STATES	10	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1)
\$CODES	1398	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
. ABS .	0	NOVEC,NOWRT,NORD ,NOEXE,NOSHR, LCL, ABS, CON,NOPIC,ALIGN(0)

Library Statistics

File	----- Symbols -----			Pages Mapped	Processing Time
	Total	Loaded	Percent		
-\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	69	0	1000	00:01.8
-\$255\$DUA28:[SYSLIB]TPAMAC.L32;1	42	19	45	14	00:00.2

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:RUNDET/OBJ=OBJ\$:RUNDET MSRC\$:RUNDET/UPDATE=(ENH\$:RUNDET)

: Size: 1398 code + 1202 data bytes
: Run Time: 00:26.6
: Elapsed Time: 01:36.1
: Lines/CPU Min: 1966
: Lexemes/CPU-Min: 20041
: Memory Used: 169 pages
: Compilation Complete

0051 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SETACNTNG
LIS

RENAMEMSG
LIS

RUNDET
LIS

RUNMSG
LIS

RENAME
LIS

RUNCUTUC
LIS

SET
LIS